A new Method
for computing One-Loop Integrals

L. Brücher[1], J. Franzkowski[2] D. Kreimer[3]

Institut für Physik Dept. of Physics
Johannes Gutenberg-Universität Univ. of Tasmania
Staudingerweg 7 G.P.O. Box 252C, Hobart 7001
D-55099 Mainz Australia
Germany

October 1993

Abstract

We present a new program package for calculating one-loop Feynman integrals, based on a new method avoiding Feynman parametrization and the contraction due to Passarino and Veltman. The package is calculating one-, two- and three-point functions both algebraically and numerically to all tensor cases. This program is written as a package for Maple. An additional Mathematica version is planned later.

[1] e-mail: Bruecher@vipmza.physik.uni-mainz.de
[2] e-mail: Franzkowski@vipmza.physik.uni-mainz.de
[3] e-mail: Kreimer@physvax.phys.utas.edu.au

PROGRAM SUMMARY

LONG WRITE-UP

1 Introduction

In particle physics the calculation of one- and higher loop corrections to particle processes is necessary to keep track with the increasing accuracy of particle colliders. In this context it is very useful if a fast and automatized calculation of the contributing Feynman diagramms is provided by a computer. On the most basic loop correction order, the one-loop level, several attempts for such programs are already available, like FF [13] – calculating scalar one-loop integrals in Fortran – or FeynCalc [12] – a Mathematica [11] package – which exhibits the usual Passarino-Veltman coefficients of tensor integrals in terms of scalar integrals. Both programs are using the standard procedure for calculating one-loop diagrams, namely Feynman parametrization, Wick rotation and the Passarino-Veltman contraction for tensor structure (cf. [8] and [9]).

Although this standard procedure became very popular during the last decade, it turned out to be rather inconvenient for being implemented on a computer if the integral involves higher tensor structure. For that reason we investigated in a new method for calculating arbitrary one-loop integrals which is already published for the two- and three-point case (cf. [3] and [4]).

Our approach is completely different from the standard procedure and the struc-

ture of other packages. We are not forced to invert big matrices which is the most difficult step in computing the tensor structure of the integrals. In our method the loop momentum integrations are performed directly, using the residue theorem and introducing $\mathcal{R}$ functions (cf. [5]), a class of special functions with some nice and useful features. Instead of big matrices we get some recursion relations for the involved $\mathcal{R}$ functions (cf. [1] or [2]), which allow a fast and memory efficient evaluation of the integrals under consideration.

The program is designed to reduce the $\mathcal{R}$ functions to a set of fundamental functions which are rewritten in terms of logarithms and dilogarithms. Up to now the different one-, two- and three-point functions are programmed as procedures in Maple [10]. Therefore our results are available numerically as well as algebraically. We tested our procedures using Maple V on a VAX 4000-90 workstation.

2 Description of the method

A detailed description of the method may be found in [1] or [2]. Here we would like to give a brief introductory overview only. The calculation is based on the well-known dimensional regularization method, as described for example in [6]. Instead of introducing Feynman parameters the integral is split up into orthogonal and parallel space (cf. [6]), so that for example the two-point integral takes on the form

$$B^{(p_0 p_1)}(q, m_1, m_2) \tag{1}$$
$$= \frac{2\pi^{\frac{D-1}{2}}}{\Gamma(\frac{D-1}{2})} \int_{-\infty}^{\infty} dl_\parallel \int_0^{\infty} dl_\perp \, l_\perp^{D-2} \frac{(l_\parallel)^{p_0} (l_\perp)^{p_1}}{[(l_\parallel + q)^2 - l_\perp^2 - m_1^2 + i\varrho][l_\parallel^2 - l_\perp^2 - m_2^2 + i\varrho]}$$

Here we present the most general case, depending on one external momentum q and two arbitrary masses m_1 and m_2. The splitting of space requires a distinction between $l_\parallel$ and $l_\perp$ which we define as

$$l_\parallel = \frac{l \cdot q}{\sqrt{q^2}}$$
$$l_\perp = \sqrt{l_\parallel^2 - l^2} \tag{2}$$

In this notation $l_\parallel$ describes the component of l which is parallel to the external momentum q, whereas $l_\perp$ represents the orthogonal complement.

The integrals (1) can be reduced to a sum of $\mathcal{R}$ functions by using (A.1) with the help of partial fraction and residue theorem. The complete procedure is described in detail in [2] and [1]. The result in the two-point-case may be written according to [3]:

$$B^{(p_0 p_1)}(q, m_1, m_2)$$
$$= \frac{i\pi^{\frac{3}{2}-\varepsilon}}{\Gamma(\frac{3}{2}-\varepsilon)} \Gamma(\tfrac{1}{2}+\varepsilon-\tfrac{p_1}{2}) \Gamma(\tfrac{3}{2}-\varepsilon+\tfrac{p_1}{2}) \frac{(-1)^{\frac{p_1}{2}} (e^{-i\pi})^{-\varepsilon}}{q(-2\varepsilon + p_1 + 1)}$$
$$\left[(-1)^{p_0 - [p_0]} (\tfrac{q}{2} + A)^{1+p_0-[p_0]} B(\varepsilon - \tfrac{p_1+[p_0]}{2}, \tfrac{[p_0]+1}{2}) \right.$$
$$\left. \mathcal{R}_{-\varepsilon + \frac{p_1+[p_0]}{2}} \left(-\tfrac{p_1+1}{2} + \varepsilon, 1; -m_1^2 + i\varrho, -(\tfrac{q}{2} + A)^2 \right) \right.$$

$$+ \sum_{i=0}^{p_0} \binom{p_0}{i} (-q)^{p_0-i} \left(\tfrac{q}{2} - A\right)^{1+i-[i]} B(\varepsilon - \tfrac{p_1+[i]}{2}, \tfrac{[i]+1}{2})$$

$$\mathcal{R}_{-\varepsilon + \frac{p_1+[i]}{2}} \left(-\tfrac{p_1+1}{2} + \varepsilon, 1; -m_2^2 + i\varrho, -(\tfrac{q}{2} - A)^2 \right) \Big]$$

where $\varepsilon = \frac{4-D}{2}$ represents the usual dimensional regularization parameter. We used the abbreviations

$$A = \frac{m_2^2 - m_1^2}{2q}$$

$$[a] = \begin{cases} a, & a \text{ even} \\ a + 1, & a \text{ odd} \end{cases}$$

The result of the three-point integral is obtained by a quite similar procedure and may be found in [2] or [4].

3 Evaluation and expansion of the $\mathcal{R}$ functions

After having expressed all possible integrals in terms of $\mathcal{R}$ functions we get easy and fast computable results by applying some recurrence relations for the $\mathcal{R}$ functions under consideration.

We achieve these relations by first reducing the number of different $\mathcal{R}$ functions to a set of fundamental $\mathcal{R}$ functions. This is possible due to the already mentioned features of the $\mathcal{R}$ functions, especially the relation (A.2) and (A.3). The reduction takes place in two steps. First of all we increase the parameter of the $\mathcal{R}$ function to get it close to 0 or 1. This can easily be achieved by using formula (A.2). In the second step we reduce the index of the $\mathcal{R}$ function by virtue of a convenient formula derived from (A.3). In the case of the two point-function it has the form:

$$\mathcal{R}_t(b_1, b_2; z_1, z_2) = \frac{-1}{t + \beta - 1} \left[((t-1)z_1 z_2) \, \mathcal{R}_{t-2}(b_1, b_2; z_1, z_2) \right. \tag{3}$$
$$\left. + ((1 - t - b_1)z_1 + (1 - t - b_2)z_2) \, \mathcal{R}_{t-1}(b_1, b_2; z_1, z_2) \right]$$

Now the power of this method is obvious. Both steps mentioned above may be performed easily by computer, due to the recursive definition of the formulae (A.2) and (A.3).

Up to now we have acquired a result which is exact to all powers of ε, though we are interested only in an expansion in powers of ε. Therefore we now substitute the expansion for the fundamental $\mathcal{R}$ functions in terms of ε, resulting in simply computable functions like logarithms and dilogarithms. This is a rather cumbersome task but has to be performed only once for each N-point function. The procedure, especially its subtleties concerning analytical continuation, is described in detail in another paper [7]. For completeness we here just cite the expansion for the scalar two-point case

$$\mathcal{R}_{-\varepsilon}(-\tfrac{1}{2} + \varepsilon, 1, z_1, z_2) = \frac{\pi i}{B(\varepsilon, \tfrac{1}{2})} \frac{(z_2 - z_1)^{\frac{1}{2} - \varepsilon}}{z_2^{\frac{1}{2}}}$$

$$+ \tfrac{1}{2} z_1^{-\varepsilon} \left(\frac{z_1}{z_2} \right)^{1-\varepsilon} \left[\left(1 + \sqrt{1 - \frac{z_1}{z_2}} \right)^{-1+2\varepsilon} + \left(1 - \sqrt{1 - \frac{z_1}{z_2}} \right)^{-1+2\varepsilon} \right] + \mathcal{O}(\varepsilon^2)$$

3

The three-point integrals

$$C^{(p_0 p_1 p_2)}(q_1, q_2, m_1, m_2, m_3)$$

$$= \frac{2\pi^{\frac{D-2}{2}}}{\Gamma(\frac{D-2}{2})} \int\limits_{-\infty}^{\infty} dl_{0\parallel} \int\limits_{-\infty}^{\infty} dl_{1\parallel} \int\limits_{0}^{\infty} dl_\perp \, l_\perp^{D-3} \frac{(l_{0\parallel})^{p_0} (l_{1\parallel})^{p_1} (l_\perp)^{p_2}}{[(l_{0\parallel} + q_1)^2 - l_{1\parallel}^2 - l_\perp^2 - m_1^2 + i\varrho]}$$

$$\frac{1}{[(l_{0\parallel} + q_{20})^2 - (l_{1\parallel} + q_{21})^2 - l_\perp^2 - m_2^2 + i\varrho] \, [l_{0\parallel}^2 - l_{1\parallel}^2 - l_\perp^2 - m_3^2 + i\varrho]}$$

are calculated in a quite similar way (cf. [2] or [4]). Again we treat the general case with two independent external momenta q_1, q_2 and three masses m_1, m_2, m_3. For the momentum components we write in a quite natural generalization:

$$l_{0\parallel} = \frac{l \cdot q_1}{\sqrt{q_1^2}}$$

$$l_{1\parallel} = -\frac{l \cdot q'_2}{\sqrt{q'^2_2}}; \quad q'_2 = q_2 - \frac{q_1 \cdot q_2}{q_1^2} q_1$$

$$l_\perp = \sqrt{l_{0\parallel}^2 - l_{1\parallel}^2 - l^2} \tag{4}$$

$$q_{20} = \frac{q_1 \cdot q_2}{\sqrt{q_1^2}}$$

$$q_{21} = \sqrt{q_{20}^2 - q_2^2}$$

The evaluation is resulting in the fundamental $\mathcal{R}$ functions

$$\mathcal{R}_{-2\varepsilon}(\varepsilon, \varepsilon, 1; x, y, z) = z^{-2\varepsilon} \left\{ 1 + 2\varepsilon^2 \left[\mathrm{Li}_2\left(1 - \frac{x}{z}\right) + \mathrm{Li}_2\left(1 - \frac{y}{z}\right) \right. \right.$$

$$+ \ln\left(1 - \frac{x}{z}\right) \eta\left(x, \frac{1}{z}\right) + \ln\left(1 - \frac{y}{z}\right) \eta\left(y, \frac{1}{z}\right) + \ln z \left[\eta\left(x - z, \frac{1}{1-z}\right) \right.$$

$$\left. \left. \left. - \eta\left(x - z, -\frac{1}{z}\right) + \eta\left(y - z, \frac{1}{1-z}\right) - \eta\left(y - z, -\frac{1}{z}\right) \right] \right] \right\} + O(\varepsilon^3)$$

$$\mathcal{R}_{1-2\varepsilon}(\varepsilon, \varepsilon, 1; x, y, z) = (1 - 2\varepsilon)z^{1-2\varepsilon} + \varepsilon(x + y)$$

$$+ 2\varepsilon^2 \left[-y \ln y - x \ln x + z \, \mathrm{Li}_2\left(1 - \frac{x}{z}\right) + z \, \mathrm{Li}_2\left(1 - \frac{y}{z}\right) \right. \tag{5}$$

$$+ z \ln\left(1 - \frac{x}{z}\right) \eta\left(x, \frac{1}{z}\right) + z \ln\left(1 - \frac{y}{z}\right) \eta\left(y, \frac{1}{z}\right) + z \ln z \left[\eta\left(x - z, \frac{1}{1-z}\right) \right.$$

$$\left. \left. - \eta\left(x - z, -\frac{1}{z}\right) + \eta\left(y - z, \frac{1}{1-z}\right) - \eta\left(y - z, -\frac{1}{z}\right) \right] \right] + O(\varepsilon^3)$$

noting that our relations are valid in all kinematical regions, so that the results of [7] apply to all the cases of our program. For a detailed discussion see [7].

4 The one-loop functions

Our aim is to calculate one-loop integrals both algebraically and numerically. Therefore the algorithm is implemented in Maple [10], a 'language' for symbolic computations. To make the implementation for any user most comfortable, the procedures

are bound to a package, which may be read from the Maple system during run-time. The user-interface consists of three elementary functions

- OneLoop1Pt(p, m)

- OneLoop2Pt(p_0, p_1, q, m_1, m_2)

- OneLoop3Pt$(p_0, p_1, p_2, q_1, q_{20}, q_{21}, m_1, m_2, m_3)$

The functions directly correspond to the following integrals

- One-point function:

$$A^{(p)}(m) = \int d^D l \, \frac{(l)^p}{[l^2 - m^2 + i\varrho]}$$

- Two-point function:

$$B^{(p_0 p_1)}(q, m_1, m_2)$$
$$= \frac{2\pi^{\frac{D-1}{2}}}{\Gamma(\frac{D-1}{2})} \int\limits_{-\infty}^{\infty} dl_\parallel \int\limits_{0}^{\infty} dl_\perp \, l_\perp^{D-2} \frac{(l_\parallel)^{p_0} (l_\perp)^{p_1}}{[(l_\parallel + q)^2 - l_\perp^2 - m_1^2 + i\varrho] [l_\parallel^2 - l_\perp^2 - m_2^2 + i\varrho]}$$

- Three-point function:

$$C^{(p_0 p_1 p_2)}(q_1, q_2, m_1, m_2, m_3)$$
$$= \frac{2\pi^{\frac{D-2}{2}}}{\Gamma(\frac{D-2}{2})} \int\limits_{-\infty}^{\infty} dl_{0\parallel} \int\limits_{-\infty}^{\infty} dl_{1\parallel} \int\limits_{0}^{\infty} dl_\perp \, l_\perp^{D-3} \frac{(l_{0\parallel})^{p_0} (l_{1\parallel})^{p_1} (l_\perp)^{p_2}}{[(l_{0\parallel} + q_1)^2 - l_{1\parallel}^2 - l_\perp^2 - m_1^2 + i\varrho]}$$
$$\frac{1}{[(l_{0\parallel} + q_{20})^2 - (l_{1\parallel} + q_{21})^2 - l_\perp^2 - m_2^2 + i\varrho] [l_{0\parallel}^2 - l_{1\parallel}^2 - l_\perp^2 - m_3^2 + i\varrho]}$$

Again we follow our notation which distinguishes between parallel and orthogonal space. The masses are denoted by m_i and the definitions of (2) and (4) for the momenta are of course still valid. We would like to emphasize the fact that in our notation the indices p_i represent the powers of the different components of the loop momentum l and should not be mixed with Lorentz indices of the corresponding tensor. This notation is perhaps not very often used, but it corresponds directly to our method of integration. The usual notation may be recovered by an additional algorithm described in section 5.

The results of the one-loop functions are Laurent expansions in terms of the ultraviolet regulator ε. The output of the described procedures consists of a list where the significant coefficients of this expansion are denoted.

input	output
OneLoop1Pt(p, m)	$[\varepsilon^{-1}\text{-}term, \varepsilon^0\text{-}term]$
OneLoop2Pt(p_0, p_1, q, m_1, m_2)	$[\varepsilon^{-1}\text{-}term, \varepsilon^0\text{-}term]$
OneLoop3Pt$(p_0, p_1, p_2, q_1,$	$[\varepsilon^{-2}\text{-}term, \varepsilon^{-1}\text{-}term,$
$\quad q_{20}, q_{21}, m_1, m_2, m_3)$	$\varepsilon^0\text{-}term, \{abb.\}]$

In the output "ε^{-1}-term" is meant to represent the divergent part (the coefficient of $\frac{1}{\varepsilon}$) whereas "ε^0-term" is describing the finite part. Of course the ε^{-2}-term has to vanish in any case, but is left in order to check the accuracy of the program in numerical calculations. In the three-point case "abb." is written for some abbreviations which are introduced to get a more compact result. For the same reason we define a function Λ which collects the ocurring η functions in the three-point case:

$$\Lambda(x,y) = \ln(y)\left[\eta\left(x-y,\frac{1}{1-y}\right) - \eta\left(x-y,-\frac{1}{y}\right)\right] + \ln\left(1-\frac{x}{y}\right)\eta\left(x,\frac{1}{y}\right) \quad (6)$$

Up to now the program is not regulating the infrared divergence of the three-point function – which may occur if one mass is set to 0 – by virtue of an infrared dimension parameter ε_{IR}. Therefore the user is expected to avoid this kind of divergence by introducing a (small) mass parameter instead of 0.

Another difficulty arises if the external momenta are collinear, which means that $q_{21} = 0$. Although this case is simpler to calculate, because the parallel space is only one-dimensional, the program is not taught to deal with this special case up to now. The user is asked to use a small q_{21}-component instead.

As long as the arguments of the OneLoop functions are symbols the output is given algebraically, whereas of course numbers inserted for the arguments imply a numerical result. Moreover, in the numerical case, the program sets the value of ϱ, the imaginary part of the propagators, five digits higher than the numerical accuracy of the whole calculation. In the algebraic case just 'rho' is returned.

For different calculations of the same OneLoop function it is of course rather inconvenient to start the whole program again. Therefore our package includes the functions OneLoopLib1Pt(n), OneLoopLib2Pt(n) and OneLoopLib3Pt(n) which generate a library of all OneLoop1Pt, OneLoop2Pt and OneLoop3Pt functions respectively up to the tensor rank n. Each function is written to one file. The filenames correspond to the integrals in the following way:

ONE0.RES	OneLoop1Pt($0, m$)
TWO00.RES	OneLoop2Pt($0, 0, q, m_1, m_2$)
TWO10.RES	OneLoop2Pt($1, 0, q, m_1, m_2$)
THREE000.RES	OneLoop3Pt($0, 0, 0, q_1, q_{20}, q_{21}, m_1, m_2, m_3$)
THREE100.RES	OneLoop3Pt($1, 0, 0, q_1, q_{20}, q_{21}, m_1, m_2, m_3$)
THREE010.RES	OneLoop3Pt($0, 1, 0, q_1, q_{20}, q_{21}, m_1, m_2, m_3$)
$\cdots$	$\cdots$

To specify the path of the library the variable "tensorpath" may be assigned to the desired directory.

5   The Passarino-Veltman coefficients

To receive a more familiar representation of the output a procedure is added which extracts the Passarino-Veltmann coefficients from our programs, PassVelt1Pt, PassVelt2Pt and PassVelt3Pt for the different n-point cases. The following types of arguments – here demonstrated for instance for the two-point function – are possible:

- PassVelt2Pt(n): The tensor decomposition of the rank M tensor two-point function

$$\int d^D l \, \frac{l_{\mu_1} \cdots l_{\mu_M}}{[(l+q)^2 - m_1^2 + i\varrho] [l^2 - m_2^2 + i\varrho]}$$

is given. The procedure returns a list consisting in the different coefficients, which are expressed in terms of the OneLoop tensor integrals, and the defining equation for the coefficients, for instance in the case $M = 2$:

$$\left[C_{21} = -\frac{\text{OneLoop2Pt}(0,2)}{3 - 2\varepsilon} \, , \, C_{20} = -\frac{\text{OneLoop2Pt}(0,2)}{q^2 \, (-3 + 2\varepsilon)} \right.$$
$$\left. + \frac{\text{OneLoop2Pt}(2,0)}{q^2} \, , \, C_{20} q_{\mu_1} q_{\mu_2} + C_{21} g_{\mu_1 \mu_2} \right]$$

- PassVelt2Pt(n,full): The function returns the same as above, but inserts the results of the OneLoop tensor integrals explicitly.

- PassVelt2Pt(n, q, m_1, m_2): The function returns the same as PassVelt2Pt(n,full), but expressed in the user defined terms for q, m_1 and m_2, using not the default parameters. Of course numerical values are also possible.

In the three-point case a similar decomposition of the rank M tensor three-point function

$$\int d^D l \, \frac{l_{\mu_1} \cdots l_{\mu_M}}{[(l+q_1)^2 - m_1^2 + i\varrho] [(l+q_2)^2 - m_2^2 + i\varrho] [l^2 - m_3^2 + i\varrho]}$$

is returned

If the variable "tensorpath" is set to some path, the PassVelt procedure assumes that the OneLoop functions are written there as described in the previous section. It then reads the necessary integrals from this directory. Otherwise, if "tensorpath" is not assigned, PassVelt is forced to calculate the necessary OneLoop functions by its own.

6 An example

To demonstrate how our package works we illustrate the calculation of the electron self-energy in QED. Starting with QED Feynman rules we get:

$$
\begin{aligned}
\Sigma(q) &= -ie^2 \mu^{4-D} \int \frac{d^D l}{(2\pi)^D} \, \gamma_\mu \frac{1}{\slashed{q} + \slashed{l} - m} \gamma_\nu \frac{g^{\mu\nu}}{l^2} \\
&= -ie^2 \mu^{4-D} \int \frac{d^D l}{(2\pi)^D} \frac{\gamma_\mu(\slashed{q} + \slashed{l} + m)\gamma^\mu}{[(l+q)^2 - m^2] \, l^2} \\
&= -ie^2 \mu^{4-D} \int \frac{d^D l}{(2\pi)^D} \, \frac{1}{[(l+q)^2 - m^2] \, l^2} \left[\gamma_\mu l_\nu \gamma^\nu \gamma^\mu + \gamma_\mu q_\nu \gamma^\nu \gamma^\mu + m \gamma_\mu \gamma^\mu \right] \\
&= -ie^2 \mu^{4-D} \int \frac{d^D l}{(2\pi)^D} \, \frac{1}{[(l+q)^2 - m^2] \, l^2} \left[l_\nu \gamma^\nu (2 - D) + q_\nu \gamma^\nu (2 - D) + mD \right]
\end{aligned}
$$

Now we split into orthogonal and parallel space noting that the γ matrix vector is split into $\gamma_\parallel$ and $\gamma_\perp$:

$$
\begin{aligned}
\Sigma(q) \;=\; & -\frac{ie^2\mu^{4-D}}{(2\pi)^D}\left\{(2-D)\left[\gamma_\parallel\cdot\int d^Dl\,\frac{l_\parallel}{[(l+q)^2-m^2]\,l^2}\right.\right. \\
& \left.-\gamma_\perp\cdot\int d^Dl\,\frac{l_\perp}{[(l+q)^2-m^2]\,l^2}\right] \\
& \left.+[(2-D)q\cdot\gamma_\parallel+Dm]\int d^Dl\,\frac{1}{[(l+q)^2-m^2]\,l^2}\right\} \\
\;=\; & -\frac{ie^2\mu^{4-D}}{(2\pi)^D}\left\{(2-D)\left[\gamma_\parallel\cdot B^{(10)}(q,m,0)-\gamma_\perp\cdot B^{(01)}(q,m,0)\right]\right. \\
& \left.+[(2-D)q\cdot\gamma_\parallel+Dm]\,B^{(00)}(q,m,0)\right\}
\end{aligned}
\tag{7}
$$

At this stage of calculation we now take the opportunity to incorporate our procedures which solve the ocurring integrals $B^{(p_0,p_1)}$. To be specific we have to compute the functions OneLoop2Pt$(1,0,q,m,0)$, OneLoop2Pt$(0,1,q,m,0)$ and OneLoop2Pt$(0,0,q,m,0)$:

$$
\text{OneLoop2Pt}(1,0,q,m,0)\equiv B^{(10)}(q,m,0)\;=\;-\frac{i\pi^2}{2\varepsilon}\,q\;+\;\frac{\pi^2}{4q^3}\Bigg\{
$$

$$
i\ln\left(1+\frac{\sqrt{(q^2-m^2)^2+4\,i\rho\,q^2}}{q^2+m^2}\right)\left[m^4-q^4+(q^2-m^2)\sqrt{(q^2-m^2)^2+4\,i\rho\,q^2}\right]
$$

$$
+i\ln\left(1-\frac{\sqrt{(q^2-m^2)^2+4\,i\rho\,q^2}}{q^2+m^2}\right)\left[m^4-q^4-(q^2-m^2)\sqrt{(q^2-m^2)^2+4\,i\rho\,q^2}\right]
$$

$$
+i(m^4-q^4)\ln\left(-\frac{(q^2+m^2)^2}{4\,q^2}\right)+2\,i\ln(-m^2+i\rho)(q^4+m^2q^2-m^4)
$$

$$
+i\,(q^2-m^2)^2\ln\left(-\frac{(q^2-m^2)^2}{4\,q^2}\right)+2\,i(q^2-m^2)^2\ln(2)
$$

$$
-\pi\left(2q^4-(q^2-m^2)\sqrt{(q^2-m^2)^2+4\,i\rho\,q^2}\right)-2\,iq^2(q^2-m^2)
$$

$$
+2\,iq^4\gamma+2\,iq^4\ln(\pi)\Bigg\}
$$

$$
\text{OneLoop2Pt}(0,1,q,m,0)\equiv B^{(01)}(q,m,0)\;=\;0
$$

$$
\text{OneLoop2Pt}(0,0,q,m,0)\equiv B^{(00)}(q,m,0)\;=\;\frac{i\pi^2}{\varepsilon}\;+\;\frac{\pi^2}{2q^2}\Bigg\{2\,i\ln(2)(m^2-q^2)
$$

$$
+i\ln\left(1-\frac{\sqrt{(q^2-m^2)^2+4\,i\rho\,q^2}}{q^2+m^2}\right)\left(q^2+m^2+\sqrt{(q^2-m^2)^2+4\,i\rho\,q^2}\right)
$$

$$
+i\ln\left(1+\frac{\sqrt{(q^2-m^2)^2+4\,i\rho\,q^2}}{q^2+m^2}\right)\left(q^2+m^2-\sqrt{(q^2-m^2)^2+4\,i\rho\,q^2}\right)
$$

$$
+i\ln\left(-\frac{(q^2+m^2)^2}{4\,q^2}\right)\left(q^2+m^2\right)-2\,i\ln(-m^2+i\rho)\left(q^2+m^2\right)
$$

$$
+i\ln\left(-\frac{(q^2-m^2)^2}{4\,q^2}\right)\left(-q^2+m^2\right)+\pi\left(\sqrt{(q^2-m^2)^2+4\,i\rho\,q^2}-2\,q^2\right)
$$

$$-2\,i\,q^2\left(\ln(\pi)+\gamma_E-2\right)\Bigg\}$$

Inserting these functions and simplifying the expression one may rewrite the result in the following way:

$$
\begin{aligned}
\Sigma(q) &= \frac{e^2}{16\pi^2}\Bigg\{\left(-q\cdot\gamma_\parallel+4m\right)\left[\frac{1}{\varepsilon}+\ln 4\pi-\gamma+\ln\left(\frac{\mu^2}{m^2}\right)\right]\\
&\quad -q\cdot\gamma_\parallel\left[\left(\frac{m^4}{q^4}-1\right)\ln\left(1-\frac{q^2}{m^2}\right)+1+\frac{m^2}{q^2}\right]\\
&\quad -4m\left[\left(1-\frac{m^2}{q^2}\right)\ln\left(1-\frac{q^2}{m^2}\right)-\frac{3}{2}\right]\Bigg\}+\mathcal{O}(\varepsilon)\\
&= \frac{e^2}{16\pi^2}\Bigg\{\left(-\slashed{q}+4m\right)\left[\frac{1}{\varepsilon}+\ln 4\pi-\gamma+\ln\left(\frac{\mu^2}{m^2}\right)\right]\\
&\quad -\slashed{q}\left[\left(\frac{m^4}{q^4}-1\right)\ln\left(1-\frac{q^2}{m^2}\right)+1+\frac{m^2}{q^2}\right]\\
&\quad -4m\left[\left(1-\frac{m^2}{q^2}\right)\ln\left(1-\frac{q^2}{m^2}\right)-\frac{3}{2}\right]\Bigg\}+\mathcal{O}(\varepsilon)
\end{aligned}
\tag{8}
$$

which is the well-known result for the electron self-energy.

7   Conclusion and further projects

Our aim was to introduce a program package that gives any computer user the possibilty to calculate one-loop particle processes without much effort in manpower and hardware. At this stage of development the program is understood to be the first step of a larger project. We do not claim that the package is up to now completely bug-free and optimized. In fact we are planning to incorporate the possibility to regulate infrared divergences also in the dimensional regularization scheme and taking into account higher powers of propagators as well as the special case of collinear external momenta.

As a further project we plan to extend this package to the four- and five-point function. Moreover we want to export our program package to Mathematica [11], because from our point of view Mathematica at the moment is the more common language than Maple. Planned is also a menu-driven user interface. But this might be rather difficult, because we haven't yet found a convenient C-language package for algebraic calculations. (For hints contact L. Brücher)

Acknowledgements

Appendix

Conventions

$D = 4 - 2\varepsilon$ dimension in the dimensional regularisation method

$\mathcal{R}_t(b; z)$ $\mathcal{R}$ function, t is called index,

             $b = (b_1, \ldots, b_n)$ parameters, $z = (z_1, \ldots, z_n)$ arguments

$\beta = \sum_{i=1}^n b_i$ sum of parameters of an $\mathcal{R}$ function

Some fundamental properties of $\mathcal{R}$ functions

- formula for integrals

$$\int_0^\infty x^{\alpha-1} \prod_{i=1}^k (z_i + w_i x)^{-b_i}\, dx = B(\beta - \alpha, \alpha)\, \mathcal{R}_{\alpha-\beta}(\mathbf{b}, \frac{\mathbf{z}}{\mathbf{w}}) \prod_{i=1}^k w_i^{-b_i} \qquad \text{(A.1)}$$

- increasing of parameters

$$\beta\, \mathcal{R}_t(\mathbf{b}, \mathbf{z}) = (\beta + t)\, \mathcal{R}_t(\mathbf{b} + e_i, \mathbf{z}) - t z_i\, \mathcal{R}_{t-1}(\mathbf{b} + e_i, \mathbf{z}) \qquad \text{(A.2)}$$

- associated functions (k is the number of parameters)

$$\sum_{i=0}^k A_i^{(k)}(t, \mathbf{b}, \mathbf{z}) \mathcal{R}_{-t-i}(\mathbf{b}, \mathbf{z}) = 0 \qquad \text{(A.3)}$$

with

$$A_i^{(k)}(t, \mathbf{b}, \mathbf{z}) = \frac{(t, i)(\beta - t - k, k - i)}{t(\beta - t - k)}(t + i - \sum_{j=1}^k b_j z_j \frac{\partial}{\partial z_j}) E_i^{(k)}(z) \qquad \text{(A.4)}$$

$E_i^{(k)}(z)$ is the elementary symmetric function, gained from

$$(1 + x z_1) \cdots (1 + x z_k) = \sum_{i=0}^k x^i E_i^{(k)}(z) \qquad \text{(A.5)}$$

References

[1] D. Kreimer. Dissertation. Univ. Mainz (1992)

[2] J. Franzkowski. Diploma thesis. Univ. Mainz (1992)

[3] D. Kreimer. Z. Phys. C54 (1992) 667

[4] D. Kreimer. Int. J. Mod. Phys. A8 No.10 (1993) 1797

[5] B. C. Carlson: Special Functions of Applied Mathematics
Academic Press (1977)

[6] J. Collins: Renormalization
Cambridge University press (1984)

[7] L. Brücher, J. Franzkowski, D. Kreimer.
Univ. Mainz preprint MZ-TH 93-11

[8] G. 't Hooft, M. Veltman. Nucl. Phys. B153 (1979) 365

[9] G. Passarino, M. Veltman. Nucl. Phys. B160 (1979) 151

[10] B. W. Char, K. O. Geddes, G. H. Gonnet, B. L. Leong,
M. B. Monagan, S. M. Watt: Maple V
Springer (1991)

[11] S. Wolfram: Mathematica
Addison-Wesley (1988)

[12] R. Mertig: Guide to FeynCalc
Univ. Würzburg (1992)

[13] G. J. van Oldenborgh, J. A. M. Vermaseren. Z. Phys. C46 (1990) 425